Realizarea unui website si lansarea acestuia in productie

Django & Docker

Autor: BIRLADEANU Alexandru-Gabriel

Grupa: 30641

Profesor: Ciprian Pavel Oprisa

Cuprins

[1. Prima etapa 3](#_Toc124370747)

[2. A doua etapa 4](#_Toc124370748)

[3. A treia etapa 6](#_Toc124370749)

# Prima etapa

Acest proiect presupune crearea unei aplicatii web de tip chat, asemanatoare cu Whatsapp Web, in care userii se pot autentifica si pot trimite mesaje text altor utilizatori. Pentru acest website se va folosi framework-ul Django, care se poate instala ca o biblioteca pentru Python.

Comenzile necesare pentru crearea unei aplicatii Django sunt urmatoarele:

python -m pip install django

django-admin startproject <project-name>

django-admin startapp <appname>

python manage.py makemigrations

python manage.py migrate

python manage.py createsuperuser

Cu ajutorul acestor comenzi vom instala libraria django, vom crea proiectul si 2 aplicatiii, respective aso\_app si members. Members va fi folosita pentru a realiza functionalitatile de logare si inregistrare a utilizatorilor, iar aso\_app va implementa toate celelalte functionalitati. Comenzile pentru efecturarea migrarilor vor trebui rulate de fiecare data cand se fac modificari la baza de date si comanda createsuperuser se foloseste pentru crearea de useri de tip admin.

Pentru pornirea serverului se va rula comanda de mai jos, dupa care va fi vizibila pagina de pornire a site-ului, ce reprezinta templete-ul pentru toate proiectele django.

python manage.py runserver

Graphical user interface, text, application, chat or text message

Description automatically generated

# A doua etapa

Pentru a implementa autentificarea utilizatorilor, vom crea aplicatia members si vom adauga in fisierul views.py urmatoarele metode:

Text

Description automatically generated

Astfel, se va folosi API-ul existent din django pentru autentificare. De asemenea, vom crea modelul Thread in models.py din aplicatia aso\_app. Acest model va avea un unser sursa si un user destinatie si va reprezenta canalul dintre cei 2 prin care se vor trimite mesaje. De fiecare data cand un user se logheaza, se va verifica daca exista un canal intre el si toti ceilalti useri. In caz contrar, se vor crea canalele lipsa in metoda de mai jos, prezenta tot in views.py din members.

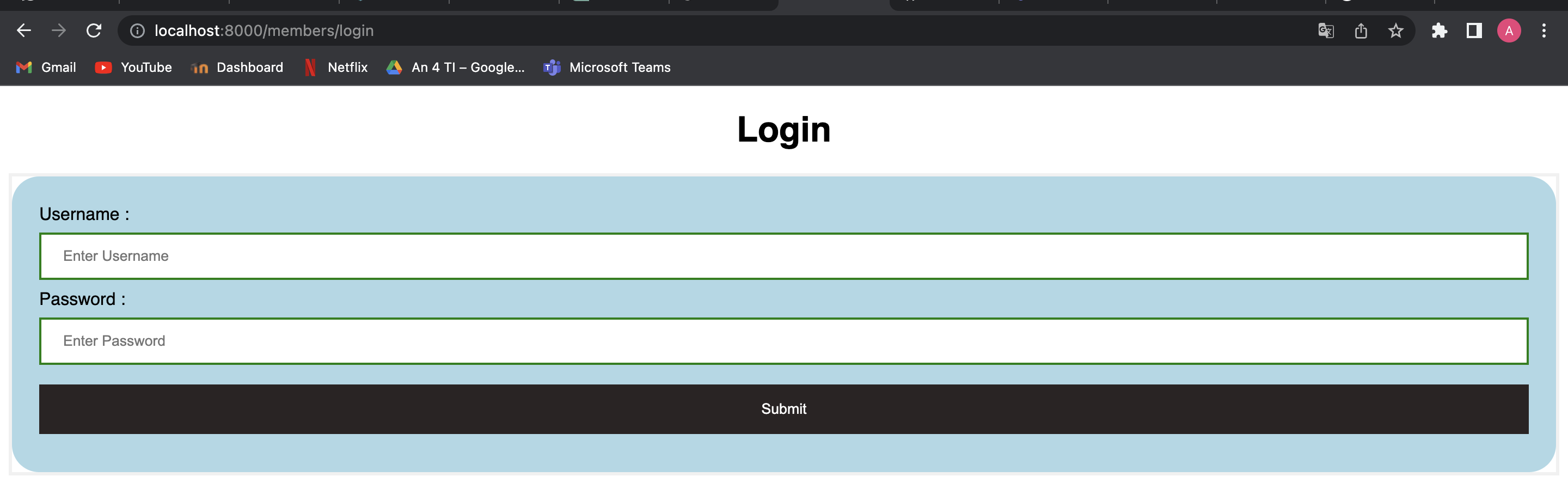
Text

Description automatically generated

Va fi nevoie de libraria channels si de ASGI pentru comunicarea asincrona si trimiterea de mesaje. In aso\_app vom crea fisierul consumers.py in care vom crea metodele asincrone pentru trimiterea de mesaje, folosind websocket-uri. Fisierul models.py va contine modelele echivalente la entitatile ce vor constitui baza de date, adica Thread, ThreadManager si ChatMessage, iar codul html pentru pagina web va fi inclus in fisierul messages.html din folderul templates creat in aceeasi locatie cu manage.py.

A screenshot of a computer

Description automatically generated with medium confidence



# A treia etapa

Pentru lansarea site-ului in productie, va fi nevoie de un container docker. Mai intai, vom adauga fisierele Dockerfile si docker-compose.yml pentru a instala dependintele necesare.

Text

Description automatically generated

Text

Description automatically generated

De asemenea , se va crea fisierul .env.dev si se vor modifica atributele din settings.py pentru a se folosi variabile din env. Text

Description automatically generated

Pentru a se migra baza de date din sqlite in postgres, se vor introduce specificatiile bazei de date in .env.dev asa cum se poate observa mai sus si se va actualiza variabila DATABASES din settings.py.

Text

Description automatically generated

La final, se vor adauga imaginea si containerul docker cu ajutorul urmatoarelor comenzi:

docker-compose up -d --build

docker-compose exec web python manage.py migrate --noinput